Self-checking logic flags errors as they happen

Parag K Lala, North Carolina Agricultural and Technical State University

Off-line testing can't detect the transient or intermittent faults that are emerging as the dominant failure mode in VLSI circuits. But self-checking circuits can continuously monitor circuits during normal operation and detect these faults.

To ensure reliability, hardware must be testable from the printed-circuit board level down to the chip level. To varying degrees, designers use techniques such as built-in self-test (BIST) to improve the testability of their products. However, BIST and other off-line test techniques are not sufficient. Failures often occur intermittently while the hardware is on line, thus circumventing these techniques. You need to incorporate self-checking circuits in your design if you want to catch logic failures while the system is on line.

Self-checking circuits have two significant advantages:

- They detect errors caused by transient or intermittent faults.
- Their fault latency is minimal; that is, the circuits detect errors immediately.

Researchers have worked considerably on making logic blocks such as adders, multipliers, and PLA device totally self-checking. They have also proposed a variety of checkers (which are themselves self-checking) for various error-detecting codes. Unfortunately, very little exists for making logic circuits in general self-checking. This article specifically discusses self-checking combinational-logic and PLA-device designs.

Self-checking circuits comprise a functional circuit having coded outputs along with a checking circuit to observe the output of the functional circuit. Fig 1 shows the model of a self-checking circuit. The functional circuit's outputs are error-detecting code words as well as being your desired logic outputs.

The checker circuit checks the validity of the output code words. It maps code-word inputs to code-word outputs and noncode-word inputs into noncode-word outputs. Thus, by observing the output of the checking circuit, you can determine if any fault exists in either the functional circuit or the checking circuit itself.

The following two definitions are the basis for self-checking logic design:

- A circuit is self-testing if, for every fault from a prescribed set of faults, it produces a noncode word at the output for at least one input code word.
- A circuit is fault secure if, for any fault from a prescribed set, it never produces a valid code word at the output for any input code word.

You can design any one of a variety of error-

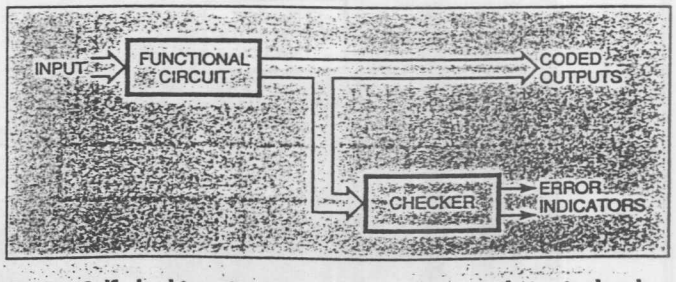

Fig 1—Self-checking circuits comprise a functional circuit that has coded outputs along with a checking circuit that observes the output of the functional circuit. By design, the functional circuit's outputs form an error-detecting code word.

where p is the number of check bits. In this article, m = 3. The main features of MOD3 residue code are:

- The check bits in a code word are only two bits.
- The residue of a bit in a code word is 1 if the bit occupies an even-bit position and 2 if the bit occupies an odd-bit position.

To illustrate, assume that the output pattern of a combinational circuit is

$$f_3f_2f_1f_0 = 1011.$$

Calculate the MOD3 residue of the pattern as

$$\underset{\text{ODD}}{\text{MOD3}}\Big[\underset{\text{EVEN}}{\text{MOD3}(f_3)}+\underset{\text{ODD}}{\text{MOD3}(f_2)}+\underset{\text{EVEN}}{\text{MOD3}(f_1)}+\text{MOD3}(f_0)\Big]$$
$$= \text{MOD3}(2+0+2+1) = 2.$$

Suppose an output bit changes its value because of a fault in the circuit. This change will increment or decrement the residue of the original output pattern by 1 or 2 depending on whether the erroneous bit occupies an even-bit or an odd-bit position, respectively. In other words, in the presence of a single-bit error, the residue will always be different from the calculated fault-free residue, thus signaling the error. For example, if bit f_1 in the above output pattern changes to 0, the new residue will be

$$\text{MOD3}(2+0+0+1) = 0.$$

Self-checking combinational logic design

To understand how you can use MOD3 residue code in self-checking combinational circuits, consider such a circuit and assume that, for a particular input pattern, Z and Z′ represent the error-free and erroneous output patterns, respectively. Further assume that $Z - Z' = -w$. If the MOD3 representation of w is not equal to 0, you can assume that a single-bit error or a multibit error is present in the output pattern.

However, a multibit error in the output pattern *can* make w a multiple of 3, making MOD3(w) equal to 0 and hence masking the presence of the error. As discussed previously, for the MOD3 residue-coding scheme, an even bit contributes a 1 to the residue, whereas an odd bit contributes a 2. Thus, only an error combination of an even and an odd bit can make MOD3(w) equal to 0.

To illustrate, assume that the error-free output of a combinational circuit for a particular input pattern is

f_5	f_4	f_3	f_2	f_1	f_0.
1	0	0	1	1	0

Suppose a fault in the circuit affects the output pattern so that both f_2 and f_1 change from 0 to 1, that is, a "unidirectional" error occurs, which yields

f_5	f_4	f_3	f_2	f_1	f_0.
1	0	0	0	0	0

Thus, because w = MOD3(38-32) = 0, the error in the output pattern is undetected. However, if you properly distribute outputs, this error becomes detectable (**Ref 5**).

The redistribution consists of changing the weights of certain bits and calculating the residue of an output pattern. For example, if you swap the positions of f_0 and f_1 (f_1 and f_0 have weight of 1 and 2, respectively), the error-free output pattern and the corresponding residue are

f_5	f_4	f_3	f_2	f_1	f_0	
1	0	0	1	0	1	= MOD3(37) = 1

In the presence of the same unidirectional error, the output pattern and the residue become

f_5	f_4	f_3	f_2	f_1	f_0	
1	0	0	0	0	0	= MOD3(32) = 2

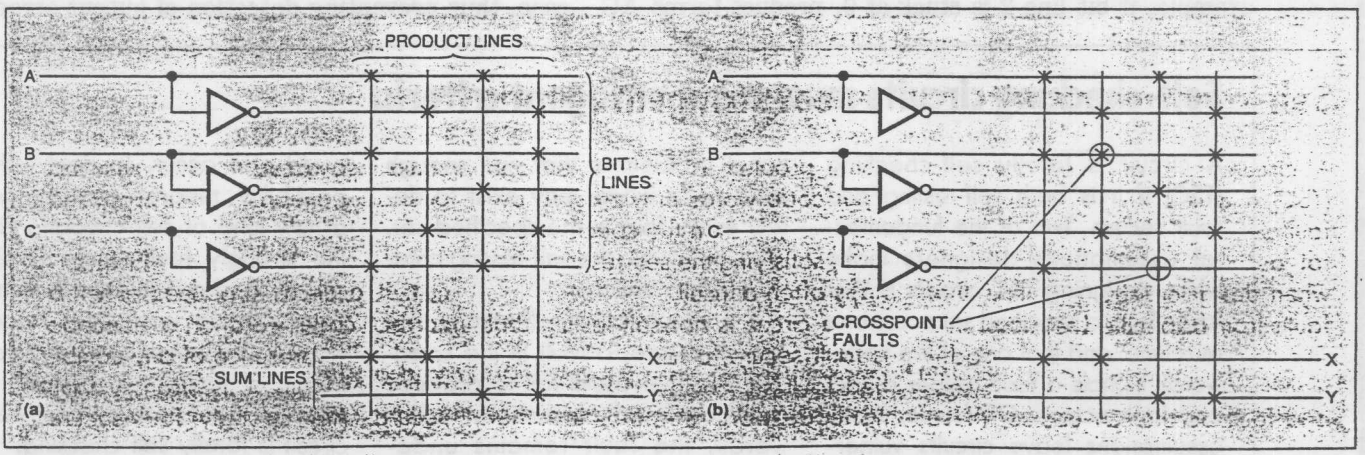

Fig 4—A PLD (a) can have failure modes other than "stuck-at" faults. Potential crosspoint failures (b) include both having a crosspoint connection where one should not be or not having a connection where one should be.

SELF-CHECKING LOGIC

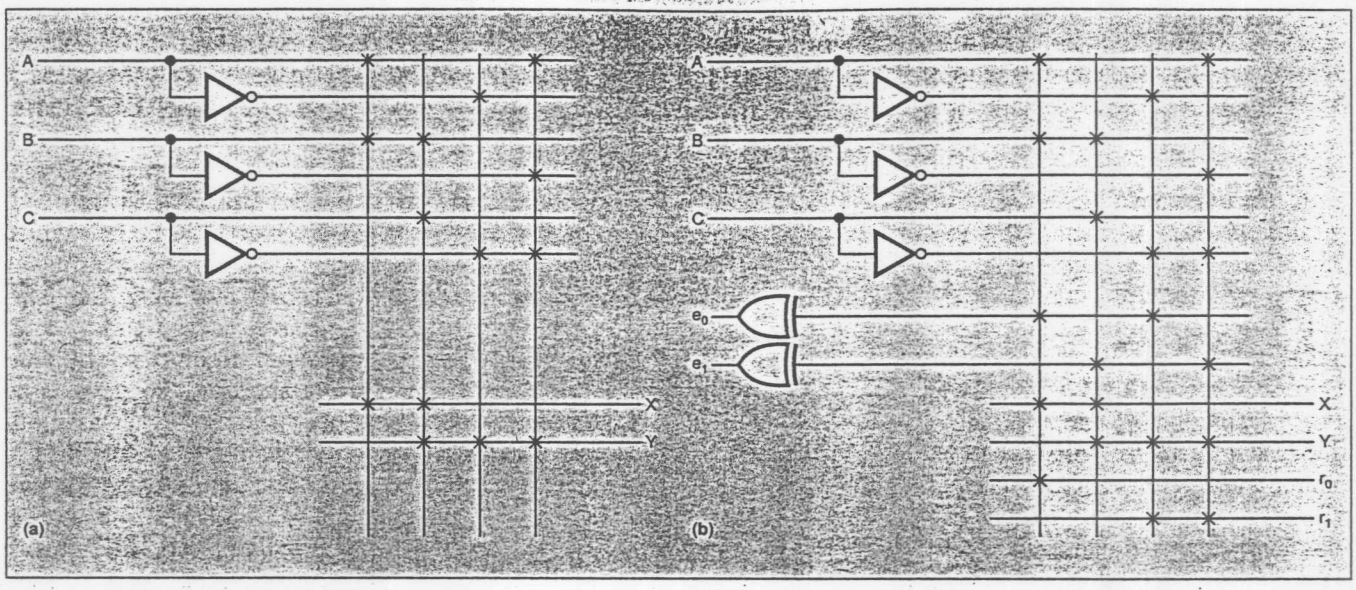

Fig 5—An arbitrary PLD (a) needs the XOR lines e_0 and e_1 as well as the residue lines r_0 and r_1 (b) to be totally self-checking.

The arbitrary PLD in **Fig 5** illustrates the self-checking technique. If the PLD satisfies the first condition specified above, the PLD will enable only one product term for an input pattern. Thus, in the presence of a fault, a desired product term may not activate, or an additional product term may activate.

However, by incorporating two XOR "trees" (**Fig 5b**), you can not only detect all single-bit errors at the output, but also detect the majority of multibit errors as well (**Ref 4**). Alternate product terms in the PLD are inputs to each XOR tree. In **Fig 5b**, if only one product term is active, the output of the XOR trees, e_0 and e_1, will be either 01 or 10. However, if none of the product terms is enabled, e_0 and e_1 will be 00. Or if an undesired product term is active in addition to the desired one, e_0 and e_1 will be 11. In either fault case, e_0 and e_1 are not a 1-out-of-2 code, which indicates a fault.

The output of the PLD uses the MOD3 residue code. Hence, you need two extra sum lines, r_0 and r_1. The MOD3 residue of the PLD's output lines, X and Y, will appear on the output lines r_0 and r_1. For example, if XY = 11 (both X and Y lines for a particular input pattern are crossed), the corresponding residue will be MOD3(3) = 0 and hence no crosses at the added sum lines will occur. In other words, $r_0r_1 = 00$.

In summary, this technique allows you to detect single faults in product terms by using XOR trees, and in sum lines from the MOD3 residue codes. Therefore, you can detect all single faults in a PLD on line. The techniques will also catch a significant portion of multibit errors. EDN

References

1. Sellers, F, M Hsiao, and L Bearnson, *Error-detecting logic for digital computers*, McGraw-Hill, New York, NY, 1968.

2. Smith, J, "Detection of faults in programmable logic arrays," IEEE Transactions, Computers, November 1979, pp 845-853.

3. Wang, S, and A Avizienis, "The design of totally self-checking circuits using PLDs," Proceedings of 9th FTC Symposium, 1979, pp 173-189.

4. Tao, D L, P K Lala, and C R Hartmann, "A concurrent testing strategy for PLDs," Proceedings of 1986 International Test Conference, pp 705-709.

5. Lala, P K, F Busaba, and K C Yarlaggada, "An approach for designing self-checking logic using residue codes," Proceedings 1990 IEEE VLSI Test Symposium, pp 166-171.

Author's biography

Parag K Lala is a research professor at North Carolina Agricultural and Technical State University in Greensboro, NC. Mr Lala has been a teacher and researcher there for four years. He obtained a M Sc (Eng) from King's College, London, UK and a PhD from the City University of London, UK. Mr Lala is a senior member of the IEEE and is also a member of the IEE. In his spare time he enjoys gardening, visiting places of historical interest, and reading Russian literature.

Article Interest Quotient (Circle One)
High 488 Medium 489 Low 490